PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/00 — A1

(11) International Publication Number: WO 99/03047

(43) International Publication Date: 21 January 1999 (21.01.99)

(21) International Application Number: PCT/US98/14292

(22) International Filing Date: 11 July 1998 (11.07.98)

(30) Priority Data:
60/052,318 11 July 1997 (11.07.97) US

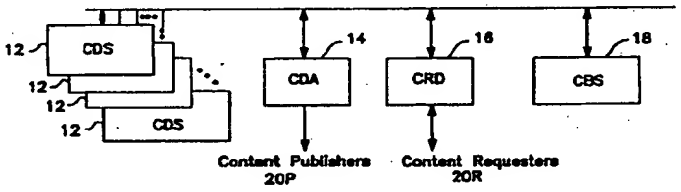
(71)(72) Applicants and Inventors: MACHARDY, Earle [US/US]; 5812 Shanna's Way, Durham, NC 27713 (US). SKARDAL, Harald [US/US]; 34 Watersedge Drive, Nashua, NH 03063 (US).

(74) Agent: CLAPP, Gary, D.; 66 Blanford Place, Bedford, NH 03110 (US).

(81) Designated States: CA, JP, US, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: CONTENT HOSTING ENVIRONMENT SYSTEM AND CACHE MECHANISM



(57) Abstract

A content hosting system that may operate as a network server for storing and publishing data according to the content, or type, of data wherein each functional element of the system and each group of cooperatively operating functional elements is designed to host. The system includes a content delivery server (12) for storing and publishing content, a content delivery administrator (14) connected to the server which manages request and content, and content dependent cache mechanism (90, 96) having a first and second cache which store content based on a predetermined size.

CONTENT HOSTING ENVIRONMENT SYSTEM AND CACHE MECHANISM

Technical Field

The present invention relates to a system for storing and operating upon data and, in particular, a system providing a content dependent environment wherein the functions and capabilities of the system and the functional elements thereof for storing and operating upon data are dependent upon and determined by the content of the data.

Background Art

A significant and primary historic trend in computer systems has been the continuous development of ever more powerful and larger general purpose processors and memories at progressively lower costs. A resultant parallel trend has been to implement all possible functions or capabilities that may be needed or desired by users on general purpose computers, with the specific functions or capabilities provided to the users being determined by the applications programs running on the general purpose computers. As a consequence, it is rare to find a system, regardless of purpose or function, that is not based upon a general purpose processor executing a general purpose, full function operating system with one or more applications programs, which are in turn generally as broadly functional as possible, providing the specific functionality desired by the user. Examples of such are as diverse as database systems, word processing and graphics creation and editing systems, environmental control and production process systems, switched communications networks, internet servers, mass data storage systems, and so on.

Such systems based upon general purpose processors and general purpose operating systems are generally regarded as advantageous because the functions and capabilities of the systems may be modified or expanded merely by adding or modifying the applications programs running on the systems. A recurring problem with such systems for many users, however, is that not all users require or are interested in the full range of functions and capabilities that may be provided by such systems, but require or are interested in only a specific, limited and well defined set of capabilities or functions.

For example, many users are interested in and need only a basic word processing program capable of creating and editing text, but without extensive document formatting capabilities, as may be used for presentations or elaborate newsletters and the like, or the capability to incorporate spreadsheets, graphics, database information, and so on, from other applications programs. One prominent New England writer, for example, is known to have written many best

selling books on a basic, single function word processing system having only the insert, overstrike, delete, copy, cut and paste functions, and is rumored to continue to write on that system. Typically, however, most single function systems are of little use for serious work. For example, most dedicated word processing systems are little more than elaborated typewriters without the functionality necessary for extensive, serious writing tasks.

As a consequence, most users are forced to accept the cost and complexity of a full function, general purpose system executing a full function, general purpose operating system, and applications programs of the corresponding complexity necessary to operate on a general purpose processor and operating system, solely to perform a single, well defined and limited function or task. It must also be noted that the cost and complexity of using a full function, general purpose system for a single task is not limited only to the cost and complexity of the hardware and software, but often includes the need for expert, highly trained operators to operate and maintain the system.

Another persistent problem in computer systems is in the various methods by which the systems store data and, most particularly, in the methods used to provide faster access to at least selected portions of the data, such as the various forms of caching mechanisms. As is well known and understood by those of ordinary skill in the relevant arts, the bulk of the data residing in a system is typically stored in relatively slow mass storage devices, such as disk drives and large but relatively slow memories. Selected portions of the data, however, are stored in a cache mechanism that is interposed between the mass storage devices and the system and that includes a relatively faster cache memory, so that the selected portions of the data are provided to the system from the faster cache memory. While caching mechanisms have been the subject of much development effort, however, the development efforts have largely been directed to methods for constructing faster caches and to methods for selecting the data to be stored in the cache memory. As a consequence, the cache mechanisms of the prior art have been generally unsatisfactory because of the inherent limitations of these approaches. That is, there are practical, physical limitations on how fast a cache memory can be constructed and only so many ways, such as the various least recently used algorithms, to select which data should reside in the cache memory.

For example, virtually every system hosts files of significantly larger size or extent than the average but which are typically accessed less frequently than are smaller files, often because of their size rather than because of their lack of significance. Such files are frequently encached by the systems of the prior art, however, because the caches are generally managed by various frequency of use algorithms and, although less frequently accessed, they are accessed frequently

enough to be encached according to the cache management algorithms commonly in use. The encaching of large files, however, often results in the effective loss of a significant part of the cache memory capacity that may be needed for caching smaller and less significant but more frequently accessed files. As a consequence, the entire system, or at least the caching mechanism, may essentially come to a halt or be significantly slowed because of the additional writing back, or flushing, and reloading of smaller, more frequently accessed files resulting from the use of cache memory capacity for very large files.

Yet another problem of the cache mechanisms of the prior art is that virtually every system includes files having contents that are essential to the operation of the system, or at least to the operations of one or more users, but which are not accessed as frequently as less critical files. In the caching mechanisms of the prior art, however, which typically include a least frequently used algorithm to determine which files are encached in the cache memory and which files are stored in a mass storage devices, such files may be "flushed" out to the mass storage device and thus not be readily accessible when needed, so that essential functions of the system are delayed while the files are retrieved to cache memory.

The present invention provides a solution to these and other problems of the prior art.

Disclosure of the Invention

In a first aspect, the present invention is directed to a content hosting system for storing and publishing data, that is, a system that provides an operating environment that is content sensitive and thereby hosts data according to the content, or type, of the data. According to the present invention, each functional element of the system and each group of cooperatively operating functional elements is designed to host, a particular content, that is, a particular type or form of data. In a system intended to operate with a single content, therefore, each functional element of the system will be designed and optimized to store and publish data of that content, and the system may be expanded to host additional contents by the addition of further functional elements corresponding to and designed and optimized for each additional content.

In a first embodiment, a content hosting system of the present invention includes at least one content delivery server for storing and publishing data of a corresponding type wherein each content delivery server is accessible to at least one content requester for providing the data stored therein to the at least one content requester, and, in systems having two or more content delivery servers, a content request director connected with each content delivery server and accessible to each content requester for identifying the locations of data in each content delivery server. According to the present invention, a content request director contains information identifying

the type of information stored in each content delivery server and the locations of data stored in each content delivery server and is responsive to a request for data from a content requester for forwarding the request to the corresponding content delivery server, so that the content delivery servers appear to a content requester as a single content delivery server containing all of the content hosted in the content hosting system.

The content hosting system also includes a content delivery administrator that is connected with each content delivery server and with the content request director, if one is present, and is accessible to at least one content publisher wherein the content delivery administrator is responsive to a request from a content publisher for a content editing operation to be performed on data stored in a content delivery server, such as storing, editing or managing the content, for providing the request to a corresponding content delivery server. The corresponding content delivery server is responsive to the content editing request for performing the requested content editing operation on the data. The content delivery administrator is also responsive to the content requester and content publisher requests for access to or operations upon the data for managing the storing of data in the content delivery servers to balance the load of requests among the content delivery servers.

In further aspects of the present invention, a content hosting system may include a content backup server connected with each content delivery server for storing a copy of the data stored in each content delivery server and for providing the stored data to the content delivery servers.

In a further embodiment of the present invention, a content hosting system operates as a network server wherein the content request director is connected from a network and to the content delivery servers for forwarding requests from content requesters connected to the network to the content delivery servers while the content delivery administrator is connected from the content publishers and to the content delivery servers for providing access to the content delivery servers to the content publishers.

In a yet further embodiment, a hierarchical content hosting system will include a plurality of content hosting systems, each including at least one content delivery server for hosting content of a corresponding type wherein each content delivery server is accessible to at least one content requester for providing the content stored therein to the at least one content requester, and, in systems having two or more content delivery servers, a content request director connected with each content delivery server and accessible to each content requester for forwarding requests from content requesters to the content delivery servers, and a content delivery administrator

connected with each content delivery server and the content request director and accessible to at least one content publisher.

The hierarchical content hosting system will generally be organized as a tree structure, with a content hosting system occupying each node of the tree wherein the content request director of each content hosting system is associated with and contains information identifying the locations of content in the associated content hosting system. At least some of the content hosting systems of a hierarchical content hosting system are capable of acting as content distributors, wherein a content distributor is capable of replicating content to the content hosting systems located on the nodes of the branches descending from the content distributor system. As such, content published in one content hosting system may be replicated from a content distributing hosting system to the content hosting systems located at the nodes of the branches descending from that content hosting system.

In another aspect, the present invention is directed to a cache mechanism for use in a data processing system, such as a content hosting system or other from of data processing system.

According to the present invention, the cache mechanism of the present invention includes a first cache for storing and providing data from files of less than a stored predetermined threshold size and a second cache for storing and providing data from files of greater than the stored predetermined threshold size. The first cache includes a first cache controller for controlling operations of the first cache and a first cache memory for storing data of files smaller than the threshold size and the second cache includes a second cache controller for controlling operations of the second cache and a second cache memory for storing data of files greater than the threshold size. The cache mechanism also includes a file filter for storing a value representing the threshold size that is responsive to data read requests for directing read requests for data of files less than the threshold size to the first cache controller and read requests for data of files greater than the threshold size to the second cache controller. The cache mechanism will generally also include a mass storage device connected from the first and second caches for storing uncached data.

In a further aspect of the cache mechanism of the present invention, the first cache memory is further organized as a cache memory for storing and providing the data of files smaller than the threshold size and a pinned memory for storing and providing the data of a pinned file wherein the pinned memory is not functionally separate from the cache memory but is effectively contiguous with the cache memory wherein the parts of the cache memory that comprise the pinned memory are identified by the state of the contents of those portions of cache

memory and wherein the data of a pinned file is locked from flushing from the pinned memory. According to this aspect of the cache mechanism, the cache mechanism further includes a cache manager for directing operations of the cache mechanism, including storing a pinning threshold value representing a threshold frequency of access of files and a cache log for recording accesses to each file having data resident in the first and second caches wherein the cache manager is responsive to the frequency of access of each file having data resident in the first and second caches for storing each file having a frequency of access greater than the threshold frequency in the pinned memory.

In a further embodiment of the cache mechanism, the pinned memory and the cache memory are designated locations in a small cache memory and a location of the small cache memory is designated as in the pinned memory by indicating that the data residing in the location is stored in a pinned table. Also, in a yet further aspect of the cache mechanism, the cache manager is responsive to the frequency of accesses of a file stored in the second cache for transferring the file into the pinned memory when the frequency of accesses of the file is greater than the threshold frequency.

Brief Description of the Drawings

The foregoing and other objects, features and advantages of the present invention will be apparent from the following description of the invention and embodiments thereof, as illustrated in the accompanying figures, wherein:

Fig. 1 is a block diagram of a content hosting system;

Fig. 2A is a block diagram of a content hosting system implemented as a network file server;

Fig. 2B is a block diagram of a content delivery server in a network file server;

Fig. 3 is a block diagram of a hierarchical content hosting system;

Fig. 4 is a block diagram of a content delivery server;

Fig. 5 is a block diagram of a content delivery administrator;

Fig. 6 is a block diagram of a content request director; and

Fig. 7 is a block diagram of a content sensitive cache mechanism.

Best Mode for Carrying Out the Invention

A. Principles of Operation

As will be described in the following, the content hosting environment system of the present invention is based on the principle that the requirements for hosting data, that is, storing, editing and managing data, such as the various forms of information contained in files, are

determined by the content of the data. In this regard, and for purposes of the present invention, content is essentially the type or form of data contained in a file, an object or other form of body of data, such as, and for example, alphanumeric text of some particular form, such as a document, a spreadsheet, or a database record, graphic information, a web page, such as a hypertext markup language (HTML) page, and so on.

The requirements for storing, editing or managing data, are essentially determined by, and limited, for each particular data content, that is, the particular type or form of data. For example, it is not necessary to provide graphics editing capabilities in a system functional element that is intended to operate on text files and it is likewise unnecessary to provide text editing functions in a system functional element that is intended for the creation and editing of bitmapped graphics. Likewise, it is not necessary to provide graphics editing capabilities in a system element that is intended only to publish the graphics data as the graphics data may be created and editing in a separate system or element.

In a like manner, the requirements for storing and retrieving data of a particular content are determined and limited, to a great extent, by the data content. For example, text files typically require less space for storing than do graphics files, so that a storage element or unit intended for text content may be designed or optimized to handle a relatively large number of smaller files while a storage element or unit could be designed to handle a smaller number of larger files.

As will be described in the following, therefore, the system of the present invention provides an operating environment that is content sensitive, that is, hosts data according to the content of the data, so that each functional element of the system and each group of cooperatively operating functional elements is designed to host, a particular content, or type or form of data. In a system intended to host a single content, therefore, each functional element of the system will be designed and optimized for data of that content, and the system may be expanded to host additional contents by the addition of further functional elements corresponding to and designed and optimized for each additional content. In this regard, it will be recognized that certain contents share common characteristics, so that at least some of the functional elements designed for and intended for use with one content will also be functional with one or more other contents.

Therefore, and according to the present invention, each functional element of the system is essentially designed as a single purpose element to host a single content and is thereby required to meet and provide only a limited and well defined set of requirements and functions, generally those functions necessary for publishing, editing and managing the content. As such, each

functional element can and typically will be simpler, less complex, of lower cost, and simpler to operate and maintain than a corresponding functional element of a full, general purpose system.

Such elements may be referred to as "appliances", wherein the term "appliance" denotes a relatively less complex and expensive functional element that is relatively easy to install and maintain and is designed to provide a relatively limited range of functions for a single content, or a set of closely related contents. According to the present invention, a user may install an appliance, or set of cooperatively operating appliances, for each specific content desired by the user, so that a system having a plurality of contents will be comprised of corresponding single content appliances.

Having described the basic principles of operation of a system according to the present invention, the following will describe an implementation or embodiment of those principles in a system, such as a internet web server. In this regard, it will be noted that the detailed design, construction and operation of many of the components of such a system will be well understood by those of ordinary skill in the relevant arts, as will various alternate implementations and embodiments of such components. As such, the following discussions will describe the components of a system according to the present invention only to the level of detail necessary for one of ordinary skill in the relevant arts to implement such a system.

B. Description of a Content Hosting Environment System

1. General Description (Fig. 1)

Referring now to Fig. 1, therein is shown a generalized block diagram of a system, hereafter referred to as a Content Hosting Environment (CHE) 10 of the present invention. Before beginning the following descriptions, it must be noted that the term "content" is used herein to refer both to the type, class or form of a body of data, such as a file or object, and to the data itself, or a body or data, thereby emphasizing, for purposes of the following descriptions, the essential correlation or unity between a body of data and its content according to the basic principles of the present invention.

As illustrated in Fig. 1, a CHE 10 is typically comprised of four types of components, or functional units. These components include one or more Content Delivery Servers (CDSs) 12 that are interconnected with a Content Delivery Administrator (CDA) 14 and, in those CHEs 10 having two or more CDSs 12, a Content Request Director (CRD) 16. A CHE 10 will usually also include at least one Content Backup Server (CBS) 18.

As will be described further below, each CDS 12 operates as a host for a corresponding content, that is, type or form of data, and stores and provides that content to Content Publishers

20P and Content Requesters 20R, which interface respectively with the CDA 14 and the CRD 16. In the general embodiment of a CHE 10 according to the present invention, therefore, a CHE 10 thereby supports two types or classes of user wherein Content Publishers 20P are generally those users permitted to add to or otherwise modify the contents of a CHE 10 while Content Requesters 20R are generally those users permitted only to read or receive content from a CHE 10, although in certain embodiments neither limitation need be enforced, or other limitations may be used. Also, while separate interfaces and capabilities for two classes of users is not necessary implemented in all embodiments of a CHE 10, this capability is particularly useful in, for example, network servers, such as World Wide Web servers. In this example, the publishers of web pages on a given CHE 10 would function within the capacity of Content Publishers 20P while web browsers accessing the CHE 10 from across the internet would function as Content Requesters 20R.

The CDA 14 therefore provides a unified interface for the Content Publishers 20P through which the Content Publishers 20P may communicate with the CDSs 12. A CDA 14 also, for example, monitors the operation of a CHE 10 to provide load balancing among the CDSs 12 by controlling and managing the distribution of content among the CDSs 12, for example, by selecting which CDS 12 is to receive and store new data and, in at least some embodiment, by transferring content between the CDSs 12.

CRDs 16, in turn, receive all incoming requests from Content Requesters 20R and, based upon information stored therein provided from the CDA 14, identify the CDSs 12 containing, or hosting, the requested content and will forward the Content Requester 20R requests to the corresponding CDSs 12. As will be described further below, the CRD 16 is not involved in the actual providing of content from a CDS 12 to a Content Requester 20R, as the requested content is, in general, provided directly from the corresponding CDSs 12 to the Content Requesters 20R.

Finally, each CBS 18 provides backup and archival services for the CHE 10.

2. A CHE 10 As A Network Server (Figs. 2A, 2B and 2C)

Referring now to Fig. 2A, therein is illustrated an exemplary CHE 10 operating as a network server connected from a Communications Network 22, such as a web server connected from the internet. As is well known and understood, the World Wide Web, usually referred to as "the Web", is constructed on the internet and a given site, or server, may have, or "host", literally thousands of hypertext markup language (HTML) pages, the location of each being identified by a corresponding universal resource locator (URLs) and each page frequently containing one or more URL's to other pages. As is well understood, each URL essentially points

to or is an address of a location of a body or source of information on the internet and contains the information required to direct a TCP/IP based protocol executing on a system to a particular location or resource on the internet and on intranets, such as a CHE 10, hosting the corresponding page.

As represented in Fig. 2A, a typical web server CHE 10 will include one or more CDSs 12, each of which is a web manager, or server, for storing and providing web pages in response to URLs received from a Content Requester 20R through Communications Network 22 and the CRD 16. In this example, and as generally illustrated in Fig. 2B, each CDS 12 may be comprised, for example, of the Netscape Navigator program executing on a Personal Computer (PC) 24 and may include a Mass Storage Device (MSD) 26 such as a disk drive or equivalent form of mass storage device, with a Controller 28 that is responsive to read and write requests including identifications of page stored or to be stored therein to read or write the corresponding pages from MSD 26 and to provide the pages to Communications Network 22. In other implementations a CDS 12 may be comprised of a dedicated server type element, generally similar to an intelligent mass storage device executing a network server program written specifically for the CDS 12, but will appear generally as illustrated in Fig. 2B.

As has been described, each CHE 10 containing two or more CDSs 12 will include a CRD 16 wherein the CRD 16 contains or stores information identifying, for the associated CHE 10, the CDSs 12 storing particular content and, for this purpose, includes a Content Map (Map) 30 identifying the CDS 12 hosting each content, such as a web page, hosted in the CHE 10. In the present example, and as illustrated in Fig. 2C, Map 30 contains an Entry 32 for and corresponding to each page hosted in a CDS 12 of the CHE 10 wherein each Entry 32 is indexed by the URL of the corresponding page stored in the CDSs 12. Each Entry 32, in turn, typically contains a CDS Identification (CDSI) 34A of the CDS 12 hosting the corresponding page. As is well understood in the art, each CDS 12 may typically store the pages in the form of files and each CDSI 34 may include a Content Location (CL) 34B identifying a location of a page in CDSs 12, for example, a file and path name and track and sector identifiers, depending upon the specific implementation of the CDSs 12 and Controllers 28.

As described, a CRD 16 will receive a request for content from a Content Requester 20R, will identify the CDS 12 in the CHE 10 hosting the requested content, and will forward the request to the proper CDS 12, which will respond to providing the requested content to the Content Requester 20R. CRDs 16 thereby cause the CDSs 12 of a CHE 10 to appear as a single, integrated CDS 12 to the Content Requesters 20R, so that Content Requesters 20R are not aware

of the internal structure or organization of a CHE 10. It will be recognized that the use of a CRD 16 in each CHE 10 containing more than one CDS 12 is of particular value in a World Wide Web type system wherein pages and portions of pages are linked together through URL type identifiers, and wherein pages or portions thereof are identified for future reference by "bookmarking", that is, the storing of URLs. In particular, all references or links to pages or content in a given CHE 10 from outside the CHE 10, such as URLs bookmarked or otherwise stored by a Content Requester 20R, essentially reference to the CHE 10's CRD 16, so that pages or content hosted in the CDSs 12 of the CHE 10 may be moved, edited, updated, added to, or otherwise modified without invalidating the references or links.

It will also be recognized by those of ordinary skill in the relevant arts that the Maps 30 residing in the CRDs 14 of CHEs 10 hosting other contents will differ in detail from that illustrated herein, but will be generally similar in structure and function. It will also be recognized by those of ordinary skill in the relevant arts that a particular content, that is, a particular body of data or file, may be comprised of multiple parts, such as a set or group of related contents, a multi-part file or set of linked files, and that the parts of a content may reside on separate CDSs 12.

As has been described, the CHE 10 also includes a CDS 14 that provides a unified interface for Content Publishers 20P wherein, in this embodiment, Content Publishers 20P are the users publishing web pages through the CHE 10. As described, the CDA 14 monitors the operation of the CHE 10, such as the storage or hosting of web pages among the CDS 12 and, in this embodiment, most probably the number of requests serviced by each CDS 12, and balances the load among the CDSs 12 by controlling and managing the distribution of content, that is, pages, among the CDSs 12. CDA 14 may do so by selecting which CDS 12 is to receive and store new data and by transferring pages between the CDSs 12 as necessary to balance the servicing of requests by the CDSs 12. As will be discussed further below, CDA 14 generates and updates an original copy of Map 30 as the pages are stored in or transferred among the CDSs 12, and provides an updated copy of the Map 30 to the CRD 16 for use by the CRD 16 in routing incoming requests from Communications Network 22. Finally, and as also described, the CBS 18 provides backup and archival services of the pages hosted by the CHE 10.

It will be recognized by those of ordinary skill in the relevant arts that a user, such as a business office, may install a number of different CHEs 10, each serving a different function or set of functions with respect to a corresponding content, or each providing a different content. For example, the office may have a web server CHE 10 as described above, but may also have

one or more CHEs 10 constructed to handle a particular text content, such as a database, or one or more graphics content CHEs 10.

3. A Hierarchical CHE 10 (Fig. 3)

Referring to Fig. 3, however, therein is illustrated an exemplary Hierarchical/Nested CHE (HCHE) 36 comprised of a plurality of hierarchical or nested CHEs 10, all of which may be constructed to host the same content or at least some of which may be constructed to host different contents. The HCHE 36 shown therein may also be geographically distributed, with the CHEs 10 being distributed on a local, regional or global level. In the instance of a web server HCHE 36 constructed according to Fig. 3, for example, the upper level CHEs 10 may function one the global level, the next tier on the regional level, and the lowest tier on the local or departmental level. This configuration will thereby support the desire of certain clients to publish on a global scale while allowing other to publish only on a local or department level.

As generally illustrated in Fig. 3, a HCHE 36 will generally be organized as a tree structure with a CHE 10 occupying each node of the tree and the hierarchical relationship between the CHEs 10, such as that between global, regional and local CHEs 10, defined according to the branches of the tree structure. At least some of the CHEs 10 of a HCHE 36 will generally be capable of acting as content distributors, wherein a content distributor CHE 10 is capable of replicating content to the subordinate CHEs 10 located on the nodes of the branches descending from the content distributor CHE 10. As such, content published in a distributor CHE 10 may be replicated from that CHE 10 to the subordinate CHEs 10 located at the nodes of the branches descending from that distributor CHE 10.

Again, the CRD 16 of each CHE 10 contains information identifying the locations of content in the associated content hosting system, so that a given Content Requester 20R will not be aware of whether it is communicating with a single CDS 12 or multiple CDSs 12. In the instance of a HCHE 36 with a pattern of distributed or replicated content defined by the hierarchical tree structure of the HCHE 36, therefore, the access that a given Content Requester 20R has to published content will be determined by the CHEs 10 to which the Content Requester 20R has access, so that different Contents Requesters 20R may be provided with different content according to the configuration of the HCHE 36 tree structure. For example, a publisher of an electronic magazine may be based in New York and may produce separate editions for the East and West coast regions of the United States. The publisher may therefore upload both versions into a global CHE 10 located in Chicago, Illinois and the two editions may be respectively replicated to regional CHEs 10 in, for example, Washington, D.C. and Los Angles,

California and, from there, to local CHEs 10 in the two regions. The local subscribers of the magazine would then access their regional editions through the local CHEs 10. If a particular subscriber located, for example, in the eastern region, wished to access the western edition, that subscriber would do so by obtaining access to a western local CHE 10.

4. CDSs 12 (Fig. 4)

Referring now to Fig. 4, therein is shown a generalized block diagram of a CDS 12. As has been described, a CDS 12 serves incoming requests by providing the requested content to the requester, whether a Content Publisher 20P or a Content Requester 20R. The context of the CDS 12, however, that is, configuration of the specific CHE 10 or HCHE 36 in which the CDS 12 resides, the hierarchical relationship of its CHE 10 to other CHEs 10, and the source of the requests is transparent to the CDS 12.

As illustrated in Fig. 4, a CDS 12 may include a Mass Storage Device (MSD) 38 for storing content and a Memory 40 for storing programs for controlling a Processor 42 for executing requests for operations with respect to or upon the content with associated Memory 40. The operations performed by Processor 42 may be limited to the storage and retrieval of content to and from MSD 38, if the CDS 12 is primarily functioning as a content storage component.

A CDS 12 operates internally according to a Configuration 44, which contains parameters describing such functions as the service level of the CHE 10, an access log, and so on, including parameters used in the general management of the operations of the CDS 12. Depending upon the specific design of the CDS 12 and the environment in which it is operating, the Configuration 44 may be provided, for example, as a separate file stored in MSD 38, a sub-tree in a Microsoft Windows NT directory, or as a file through RPC calls to an API.

Each CDS 12 will also contain and maintain a set of Log 46 files to track the request load, client trails, which content, or files, where accessed or served, and so on. The Log 46 files are generally stored internally in directories that are accessible, for example, to the CDA 14 for use in load balancing. It is apparent that a CDS 12 may also include components such as a Interface Drivers (DRVRs) 48 if, for example, the CDS 12 is required to provide content onto a network or to interface with other sources or destinations of content other than the CDA 14.

Finally, a given CDS 12 may include Authorization Mechanisms (AMs) 50, to assure that only authorized users can set or modify the Configuration 44. AMs 50 may also provide general content security by performing requester or client authorization functions.

Again, in other implementations a CDS 12 may be comprised of a dedicated server type element, generally similar to an intelligent mass storage device executing a network server program written specifically for the CDS 12, but will appear generally as illustrated in Fig. 2B.

5. CDAs 14 (Fig. 5)

Referring now to Fig. 5, a CDA 14 is generally implemented with a Processor 52 for controlling and executing CDA 14 operations and a Memory 54 for storing content and programs for controlling the operations of Processor 52 and will generally include a Mass Storage Device (MSD) 56 for storing content and the programs. A CDA 14 may also include Interface Drivers (DRVRs) 58 as necessary to physically interface with Content Publishers 20P and may include Device Interfaces (DIs) 60 as necessary to communicate with the CDSs 12, the CRD 16 and the CBS 18.

As described, it is intended in the preferred embodiment of a CHE 10 of the present invention that each CDA 14 or other element of the CHE 10 should be of limited and specific functionality specifically designed for the content to be hosted or otherwise dealt with by an element of a CHE 10, such as storing, publishing or managing the specific content. Accordingly, a CDA 14 performs three primary functions, the first of which is to provide a "front end" and interface to Content Publishers 20P and to provide and receive content to and from Content Publishers 20P, for example, through DRVRs 58. Secondly, a CDA 14 also stores a CHE Configuration 52 containing parameters defining, for example, the number, types and capacities of CDSs 12, levels of service, request loading, operational parameters, information pertaining to CRD 16, and so on.

Finally, a CDA 14 monitors the level of activity by the CDSs 12 of the CHE 10 and stores content in the CDSs 12 and transfers content between CDSs 12 in a manner to accommodate the desired request service levels. In association with this function, and as described, a CDA 14 also maintains a resident copy of the Map 30 identifying the locations of content in the CDSs 12, updating the resident Map 30 and the CRD 16 resident copy of the Map 30 as new content is added to CDSs 12 or moved between CDSs 12. The CDA 14 will also exchange content location information and configuration information with other CHEs 10 in a HCHE 36 in the same manner, updating the Map 30 and the CRD 16 resident copy of Map 30 as necessary for the routing of requests among the CHEs 10 of a HCHE 36, as has been described.

It will therefore be apparent that the function of the CDA 14 in a CHE 10 is to serve as the primary decision making and management component of the CHE 10, thereby removing all functions that are essential to the optimum functioning of the CHE 10, but which do not require

high levels of processing resources, from the request servicing components, that is, the CDSs 12 and CRD 16. The CDA 12 thereby frees the CDSs 12 and the CRD 16 to provide the maximum levels of request servicing.

The operations performed by a CDA 14 may also include the editing or modification of content and, if so, the CDA 14 will also provides the capability to create, edit or otherwise modify content. In most applications, however, content will be created and edited "off line", that is, externally to the CHE 10, and will be uploaded as necessary to the CHE 10, thereby avoiding extended intervals wherein the content of a CHE 10 is changing and the risk of providing requester with content that is changing from access to access or even during an access.

Finally, and again, in other implementations a CDA 14 may be comprised of a dedicated element designed for the specific purpose and functions necessary for its specific and limited operations, but will appear generally as illustrated in Fig. 5.

6. CRDs 16 (Fig. 6)

Referring to Fig. 6, therein is shown a generalized block diagram of a CRD 16. As described above, there will be a CRD 16 in each CHE 10 containing two or more CDSs 12 and a CRD 16 receives requests from Content Requesters 20R and directs the requests to the CDSs 12, using the Map 30 for these purposes. As represented in Fig. 6, CRD 16 is based on a Processor 62 and a Memory 64 for storing programs controlling the operations of the Processor 62 and the Map 30.

Requests 64 are received through a Request Interface (RINT) 66, which may, for example, be a network interface component, and are parsed in a Request Parser (RPARS) 68 as necessary to extract the content identification from the Request 64, with the content identification then being used to index Entries 32 of the Map 30. The corresponding location information identifying the location or locations of the requested content is read from Map 30 and concatenated as necessary with, for example, any additional Request 64 information necessary for the CDS 12 to execute the Request 64, and forwarded to the appropriate CDS 12 where the Request 64 is executed. As has been described, the requested content is returned to the Content Requester 20R directly from the CDS 12, rather than through the CRD 16, thereby freeing the CRD 16 to process the next request.

Finally with regard to CRDs 16, it should be noted that many CHEs 10 and HCHEs 36 may replicate content across two or more CDSs 12 or CHEs 10, respectively, so that a request may be served from more than one location. In such implementations, the CRDs 16 will generally also include a Session Table (Session) 72 storing a Session Entry (SE) 74 for and

corresponding to each Content Requester 20R during a moving time window, that is, over a period extending for a predetermined interval into the past. Each such SE 74 each indexed, or identified, by the Content Requester 20R's identification and will contain state information regarding a request by the corresponding Content Requester 20R during that time window, including an identification of the CDS 12 that serviced each previous request during that time window. Session 72 thereby assures that an on-going transaction will complete properly, being serviced by the same CDS 12.

Finally, and again, in other implementations a CRD 16 may be comprised of a dedicated element designed for the specific purpose and functions necessary for its specific and limited operations, but will appear generally as illustrated in Fig. 6.

7. CBSs 18

Finally, CBS 18 will not be discussed in any further detail herein as the design and operation of such back-up servers is well known to those of ordinary skill in the relevant arts and the adaptation of such designs for operation in a CHE 10 will be well understood by those of ordinary skill in the relevant arts.

Lastly, further details and discussions of an implementation of a CHE 10 of the present invention for use as an internet Web publishing server will be found in Appendix A, which contains descriptions particularly pertaining to the translation or mapping of requests into the CDS 12 locations of the requested content by CRDs 16 and the containment of scripts to the executing CDS 12.

In closing with regard to the content hosting environment as described above, it must be noted, and reiterated, that Content Delivery Servers (CDSs) 12, Content Delivery Administrators (CDAs) 14, Content Request Directors (CRDs) 16 and Content Backup Servers (CBSs) 18 are conceived and implemented, according to the present invention, as "appliances". According to the present invention, the term "appliance" denotes a relatively less complex and expensive functional element that is relatively easy to install and maintain and is designed to provide a relatively limited range of functions for a single content, or a set of closely related contents. Further according to the present invention, a user may install an appliance, or set of cooperatively operating appliances, for each specific content desired by the user, so that a system having a plurality of contents will be comprised of corresponding single content appliances.

As such, and as described, the "appliances" described above may be implemented or embodied in many alternate forms. For example, each appliance, or some appliances, may be implemented as specifically designed and constructed dedicated purpose hardware and/or

software elements. Others may be implemented as software components on individual or separate hardware systems, such as personal computers or servers, while in other instances the appliances or at least some appliances may be implemented as software programs executing on a single, shared hardware component, again such as a personal computer or a larger computer or server. Also, the selection and number of different appliances comprising a given content hosting system, and their configuration, may differ significantly from system to system, and will be determined by the needs and functions of a particular system. For example, certain systems may not include a content request director and others may include only a few content delivery servers or many content delivery servers.

C. A Cache Mechanism (Fig. 7)

As has been discussed herein above, the cache mechanism of the present invention recognizes that a recurring problem in the cache mechanisms of the prior art is that the cache mechanisms of the prior art essentially treat all encached bodies of data, whether structured in the form of content, files, objects, data blocks, other formats or organizations, as identical except for the frequency with which they are accessed. As such, the cache mechanisms of the prior art have determined which data should be encached and which should be stored in slower mass memory, and thus be slower to access, almost entirely on the frequency with which the data is used and have thus relied, for example, on least frequently used (LRU) algorithms to determine which data should be encached in the cache memory and which data should be stored in a slower mass storage device. The cache mechanisms of the prior art have, as a consequence, generally encached very large files in the same manner as smaller files because, although they are generally less frequently accessed than are smaller files, they are accessed frequently enough to be encached according to the cache management algorithms commonly in use.

The cache mechanism of the present invention, however, recognizes that the caching of data, whether in the form of files, content, objects, data blocks or other data formats, hereafter referred to generally as "files" solely for convenience and not in any form of explicit or implicit limitation on data format or structure, is advantageously also determined by other characteristics of the files than only frequency of use. For example, and as described above, virtually every system hosts files of significantly larger size or extent than the average although such files are typically seldom accessed, often because of their size rather than because of their lack of significance. Such files are frequently encached by the systems of the prior art, however, because the caches are generally managed by various frequency of use algorithms and, although less frequently accessed, they are accessed frequently enough to be encached according to the cache

management algorithms commonly in use. The encaching of large files, however, often results in the effective loss of a significant part of the cache memory capacity that may be needed for caching smaller and less significant but more frequently accessed files. As a consequence, the entire system, or at least the caching mechanism, may essentially come to a halt or be significantly slowed because of the additional writing back, or flushing, and reloading of smaller, more frequently accessed files resulting from the use of cache memory capacity for very large files.

The cache mechanism of the present invention also recognizes that the contents of at least some files is typically so significant with regard to the operations being performed by the system, or by one or more users, that such files should be immediately accessible at all times. As such, such files should not be subjected to the operation of a least recently used algorithm, but should be encached at all times, regardless of how frequently or infrequently the files are accessed.

As such, the cache mechanism of the present invention is a content sensitive cache mechanism that encaches content according to certain characteristics of the content itself, rather than only the frequency with which particular content is requested. It will be apparent from the following descriptions, however, that the cache mechanism of the present invention is useful, and may be advantageously used, in systems other than the content hosting environment of the present example. In a content hosting system such as that described herein, however, the cache mechanism of the present invention may be utilized, for example and typically, in the CDSs 12 of a CHE 10 or, in certain circumstances, in the CRDs 16, particularly in a HCHE 36 wherein at least some of the CRDs 16 host information about the content residing in one or more other CHEs 10, as described herein above.

Referring to Fig. 7, therein is shown a block diagram of a Cache Mechanism 76 of the present invention. As represented therein Cache Mechanism 76 is provided with two primary cache memory mechanisms, the first being Cache 78 which includes a Cache Controller 80, a Cache Memory 82 and, typically but not necessarily, an associated Mass Storage Device (Mass Store) 84A, such as a disk drive. The second cache mechanism is comprised of Large File Cache 86, which includes a Large File Cache Controller 88 and Large File Cache Memory 90 and, typically but not necessarily, a Mass Storage Device (Mass Store) 84B, wherein Mass Store 84A and 84B may be separate devices or a single, shared device. As indicated, in the present embodiment, Large File Cache Memory 90 is constructed of a plurality of Buffers 92 but, in alternate embodiments, may be constructed of a memory, such as Cache Memory 92, of any of a plurality of other memory structures.

As will be described further below, Cache Mechanism 76 further includes a third, additional cache mechanism embodied in a Pinned Memory 94 and which interoperates with Cache 78. As will also be described further in the following, in the presently preferred embodiment Pinned Memory 94 is not separate from Cache Memory 82 but is contiguous with and comprised of parts of Cache Memory 82, wherein the parts of Cache Memory 82 that comprise Pinned Memory 94 are identified by the state of the contents of those portions of Cache Memory 82. In other embodiments, Pinned Memory 94 may be separate from Memory 82, or a single physical Memory 96 may organized into separate areas designed as Cache Memory 82 and Pinned Memory 94, so that Cache Memory 82 and Pinned Memory 94 share Memory 96.

As indicated in Fig. 7, Cache Requests 98 for data to be read or written are received by a Cache Request Input 100, while the data to be read or written is received into or provided from Data Input/Output (I/O) 102. Data I/O 102 is in turn connected through data paths to Cache Memory 82 and Large File Cache Memory 90 and, indirectly therethrough, to Mass Storage Devices 84A and 84B, with the data paths passing through or controlled by, respectively, Cache Controller 80 and Large File Cache Controller 88.

Cache Request Input 100 passes the read/write requests to a Cache Log 102, which records all cache read/write requests for use in managing cache operations by Cache Manager 104, and to a File Filter 106. File Filter 106, in turn, examines all read requests and, in particular, compares the size of the file or other body of data to be stored with a File Size Threshold 108. File Size Threshold 108 is a value stored therein that represents a predetermined separating those files that are judged to be "large" files and all other, smaller files. File Filter 106 then, based upon the decision as to whether a particular file or other body of data is "large" or "not large" and under the direction of Cache Manager 104, directs each read request to Cache Controller 80 when the file is judged "not large" or to Large File Cache Controller 88 when the file is judged "large", while Cache Manager 104 generates or updates a corresponding entry in Cache Log 102.

It should be noted that the threshold value stored in File Size Threshold 108 may vary from system to system or may vary with time. For example, Cache Manager 104 monitors the performance of Cache Mechanism 76 through the entries in Cache Log 102 and may alter the threshold value stored in File Size Threshold 108 depending upon the performance of the mechanism, the sizes of the files or bodies of data presently typical for the system, the sizes of files most frequency accessed, the number and sizes of "large" files relative to the total number of files, and so on.

First considering the case of a "large" file, Large File Cache Controller 88 responds to a file read request from File Filter 106 by reading the data from one or more Buffers 92 of Large File Cache 90 if the data is resident in Large File Cache 90. In the present embodiment, Large File Cache 90 is a predetermined amount of private memory organized, as described above, as Buffers 92.

As is typical and commonly understood in cache mechanisms, the requested data may not be resident in the Large File Cache 90 and must thereby be read from mass storage, either directly or through Large File Cache 90. If the data is read through Large File Cache 90, or if the data is to be encached in Large File Cache 90, it is necessary to determine which of Buffers 92 is to receive the data. In the present embodiment, Large File Cache Controller 88 uses a least frequently used algorithm to determine which of Buffers 92 to re-use, if necessary. In this regard, File Filter 106 operates with Large File Cache Controller 88, and in a like manner with Cache Controller 80, to invalidate entries in Large File Cache 86 and in Cache 78 as necessary, as when data encached therein is removed or overwritten.

Also, it should be noted that in the presently preferred embodiment, Cache Manager 104 tracks the frequency of access of large files encached in Large File Cache 86, using the information stored in Cache Log 102, and may transfer a large file to Cache 78 or even into Pinned Memory 94, as described below, if a large file is accessed with sufficient frequency.

In summary, therefore, each read request is received by Cache Request Input 100 and passed to Cache Log 102, whereupon it is read by Cache Manager 104 which, in turn, determines whether the corresponding data is stored in Cache 78 or Large File Cache 86. If the request is for a large file encached in Large File Cache 86, the request is passed to File Filter 106 which in turn directs Large File Cache 86 to provide the data through Data I/O 102, whereupon Large File Cache Controller 88 reads the data from Large File Cache Memory 90 to Data I/O 102, first loading the data from Mass Storage Device 84B to Large File Cache Memory 90, if necessary.

It will be apparent, therefore, that the provision of a large file cache separate from the cache for smaller files according to the content sensitive principles of the present invention there avoids interference between these generally less frequently accessed large files and the more frequency accessed smaller files handled in Cache 78. As such, smaller files are not displaced from the cache mechanism by a relatively few larger files, or even a single large file, so that the performance of the cache mechanism as regards the more frequently accessed smaller files is enhanced.

The performance of the cache mechanism for larger files, moreover, is generally comparable to or better than that of conventional cache mechanisms as the large file cache mechanism likewise prevents small file operations from interfering with the accessing of large files. In this regard, it should be noted that Large File Cache 78 is essentially embodied as a buffer for reads from the mass storage devices containing the large files. As such, the capacity of Large File Cache 78 may typically be sufficient to contain at least a portion of a relatively large file and may be sufficient to contain a relatively small number of large files or portions thereof, and that the capacity of Large File Cache 78 may be increased as necessary or desired. It will apparent that it will be frequently necessary for Large File Cache 78 to load large files or portions thereof from mass storage. It has been found, however, that this does not adversely affect the performance of the system as the large files are accessed relatively infrequently, so that the time required for loading of large file data from mass storage is acceptable. Also, it is more frequently the case that there will be many successive accesses of a given large file, thereby allowing the prefetching of files pages in anticipation of need, and that the flushing of an entire large file to load a different file will be a relatively infrequent event.

Briefly considering the operation of the cache mechanism for smaller files, that is, files that are not "large", Cache 78 operates in the same manner as described above with regard to Large File Cache 86, except that Cache 78 encaches files, or other bodies of data, that are not judged as "large" according to the value currently stored in File Size Threshold 108 and therefore need not be described in further detail herein.

As has been discussed above, however, Cache Mechanism 76 recognizes that certain data is typically of such significance to the system or to one or more users or is accessed so frequently that the data should preferably reside in cache memory at all times to be rapidly accessible at all times. For this reason, Cache Mechanism 76 includes a Pinned Memory 94 for storing such data wherein Pinned Memory 94 is designated as "pinned" in that the data stored therein may not be flushed or transferred out, for example, to a Mass Storage Device 84, or invalidated, in the normal operation of Cache Mechanism 76, but only in a specific administrative operation outside of the normal operations of Cache Mechanism 76.

As has been described, in the presently preferred embodiment Pinned Memory 94 is not physically or organizationally separate from Cache Memory 82. Instead, Cache Memory 82 and Pinned Memory 94 are comprised of contiguous parts of Memory 96. As such, those parts of Memory 96 that comprise Cache Memory 82 and Pinned Memory 94 are identified and organized from one another dynamically and by the state of the contents of each part of Memory

96, that is, by whether the contents of a given file residing in Memory 96 are "pinned" or "unpinned". As such, it is not necessary to physically transfer data between Cache Memory 82 and Pinned Memory 94, but only to track the state of each file resident in Memory 96. Cache Manager 104 will, in general, only lock a file to "transfer" the file from Cache Memory 82 to Pinned Memory 94, and the cache will find the pinned pages when the cache defaults.

For this purpose, Cache Manager 104 maintains a Pinned Table 110 in Cache Log 102 wherein Pinned Table 110 contains a Pinned Entry 112 for and corresponding to each file or body of data that has been designated as "pinned" and thereby "residing" in Pinned Memory 94. Cache Manager 104 may generate a Pinned Entry 112, for example, when directed to do so by a user, such as a system administrator, or when the frequency of access of a particular file or body of data exceeds a threshold value stored in a Pinning Threshold 114 stored in Cache Manager 104, whereupon that data may be transferred from, for example, Cache 78 or Large File Cache 86, to Pinned Memory 94.

As has been described, in other embodiments, Pinned Memory 94 may be separate from Memory 82, or a single physical Memory 96 may organized into separate areas designed as Cache Memory 82 and Pinned Memory 94, so that Cache Memory 82 and Pinned Memory 94 share Memory 96.

Because it is the general case that the most frequently accessed files are smaller files, that is, those files other than those designated as "large" and therefore those files normally resident in Cache 78, most files that become designated as "pinned" files and "transferred" into Pinned Memory 94 will be of files from Cache 78. In addition, and as discussed above, a large file having a sufficiently high frequency of access may be designated as a "pinned" file. As such, large files having a sufficiently high frequency of access will generally first be transferred from Large File Cache 86 to Cache 78 as their frequency of access increases, so that again "transfers" of files to Pinned Memory 94 will generally be from Cache 78 rather than directly from Large File Cache 86. Again, and for this reason in the presently preferred embodiment, Cache Memory 82 and Pinned Memory 94 presently share a single Memory 96 in the preferred embodiment of Cache Mechanism 76 so that a given file or body of data may be designated as "pinned" and effectively "transferred" from Cache Memory 82 to Pinned Memory 94 solely by the generation of a corresponding entry in Pinned Table 110.

Finally, while the invention has been particularly shown and described with reference to preferred embodiments of the apparatus and methods thereof, it will be also understood by those of ordinary skill in the art that various changes, variations and modifications in form, details and

implementation may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. Therefore, it is the object of the appended claims to cover all such variation and modifications of the invention as come within the true spirit and scope of the invention.

Claims:

1. A content hosting system for storing and publishing content, comprising:

a content delivery server for storing and providing content including data of a corresponding type,

each content delivery server being accessible to a content requester for providing the content stored therein to the content requester, and

a content delivery administrator connected with each content delivery server and accessible to at least one content publisher,

the content delivery administrator being responsive to a request by a content publisher for storing content provided by the content publisher in a content delivery server to be provided to a content requester.

2. The content hosting system of claim 1, wherein:

the content hosting system includes

two or more content delivery servers, and

a content request director connected with each content delivery server and accessible to each requester for identifying the locations of content in each content delivery server,

the content request director containing information identifying the content stored in each content delivery server and being responsive to a request for content from a content requester for forwarding the request to the content delivery server hosting the requested content,

the content delivery server hosting the requested content being responsive to the request for providing the requested content to the content requester.

3. The content hosting system of claim 1, wherein:

the content delivery administrator is responsive to requests of content requesters and content publishers for managing the storing of data in the content delivery servers to balance the load of requests among the content delivery servers.

4. The content hosting system of claim 1, further comprising:

a content backup server connected with each content delivery administrator for storing a copy of the data stored in each content delivery server and for providing the stored data to the content delivery servers.

5. The content hosting system of claim 2, further comprising a network server wherein:

the content request director is connected from a network and to the content delivery servers for providing requests from content requesters connected to the network to the content delivery servers.

6. A hierarchical content hosting system for storing and operating on data, comprising:

a plurality of content hosting systems organized into a tree structure wherein each node of the tree includes a content hosting system having a hierarchical relationship defined by the tree structure, each content hosting system including

a content delivery server for storing and providing content including data of a corresponding type,

each content delivery server being accessible to a content requester for providing the content stored therein to the content requester, and

a content delivery administrator connected with each content delivery server and accessible to at least one content publisher,

the content delivery administrator being responsive to a request by a content publisher for storing content provided by the content publisher in a content delivery server to be provided to a content requester.

7. The content hosting system of claim 7, wherein:

at least one content hosting system includes

two or more content delivery servers, and

a content request director connected with each content delivery server and accessible to each requester for identifying the locations of content in each content delivery server,

the content request director containing information identifying the content stored in each content delivery server and being responsive to a request for content from a content requester for forwarding the request to the content delivery server hosting the requested content,

the content delivery server hosting the requested content being responsive to the request for providing the requested content to the content requester.

8. A cache mechanism for use in a data processing system, comprising:

a first cache for storing and providing data from files of less than a stored predetermined threshold size, including

a first cache controller for controlling operations of the first cache, and

a first cache memory for storing data of files smaller than the threshold size,

a second cache for storing and providing data from files of greater than the stored predetermined threshold size, including

a second cache controller for controlling operations of the second cache, and

a second cache memory for storing data of files greater than the threshold size, and

a file filter for storing a value representing the threshold size and responsive to data read requests for directing read requests for data of files less than the threshold size to the first cache controller and read requests for data of files greater than the threshold size to the second cache controller,

the first and second cache controllers being responsive to read requests for providing the requested data.

9. The cache mechanism of claim 8, further comprising:

a mass storage device connected from the first and second caches for storing uncached data.

10. The cache mechanism of claim 8, the first cache memory further comprising:

a cache memory for storing and providing the data of files smaller than the threshold size, and

a pinned memory for storing and providing the data of a pinned file wherein the data of a pinned file is locked from flushing from the pinned memory.

11. The cache mechanism of claim 10, further comprising:

a cache manager for directing operations of the cache mechanism, including

storing a pinning threshold value representing a threshold frequency of access of files, and

a cache log for recording accesses to each file having data resident in the first and second caches,

the cache manager being responsive to the frequency of access of each file having data resident in the first and second caches for storing each file having a frequency of access greater than the threshold frequency in the pinned memory.

12. The cache mechanism of claim 11, wherein:

the pinned memory and the cache memory are designated locations in a small cache memory, and

a location of the small cache memory is designated as in the pinned memory by indicating that the data residing in the location is stored in a pinned table.

13. The cache mechanism of claim 10, wherein:

the cache manager is responsive to the frequency of accesses of a file stored in the second cache for transferring the file into the pinned memory when the frequency of accesses of the file is greater than the threshold frequency.

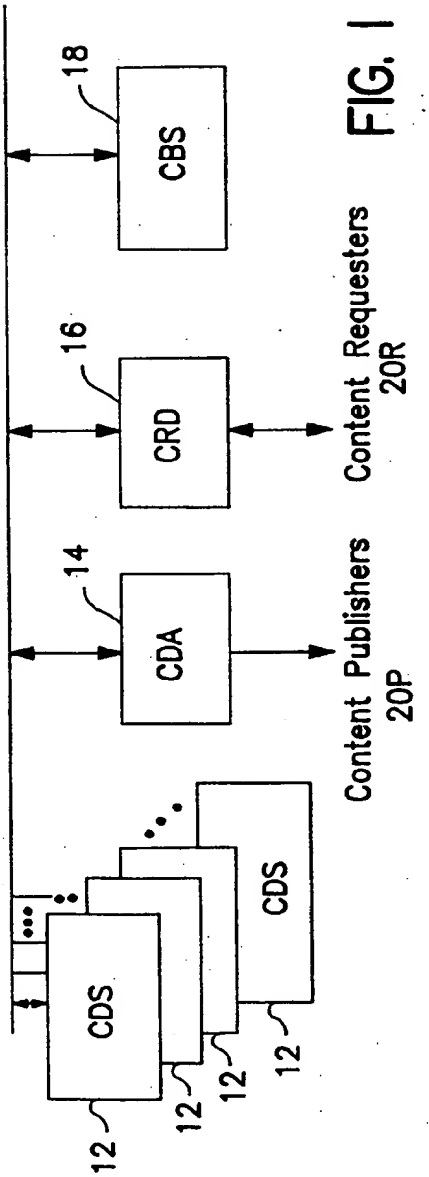

18
CBS
16
CRD
14
CDA
CDS
12
Content Requesters
20R
Content Publishers
20P

FIG. 1

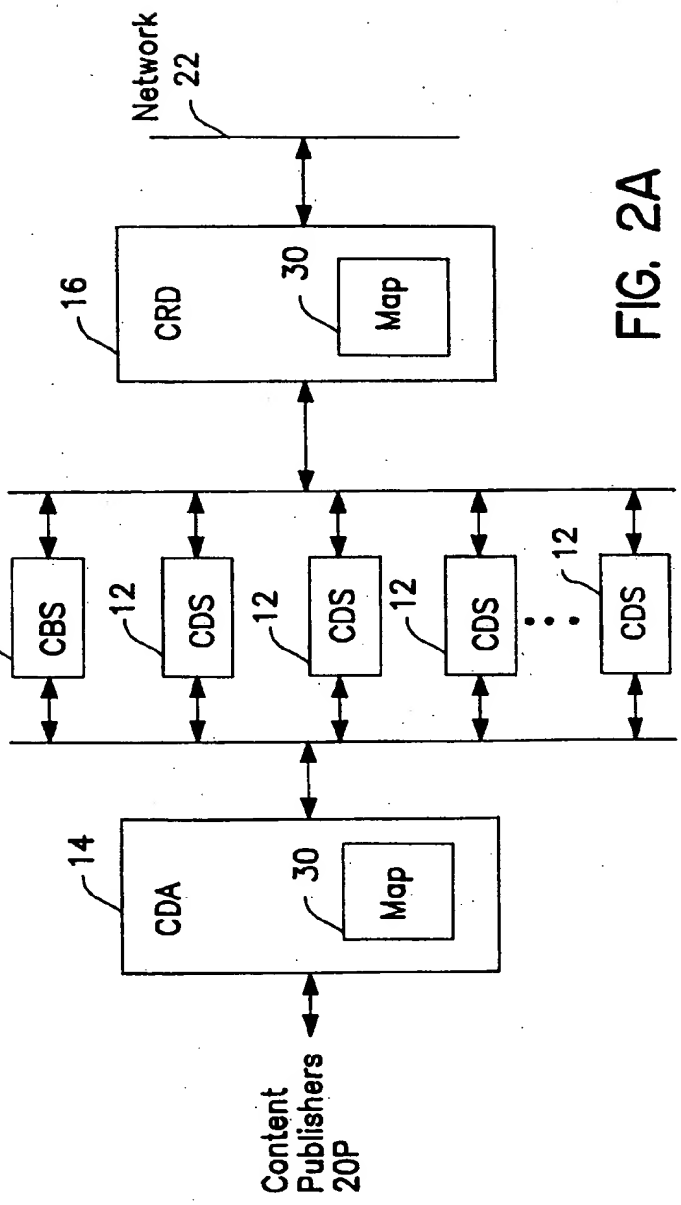

Network
22
16
CRD
30
Map
18
CBS
12
CDS
14
CDA
Content
Publishers
20P

FIG. 2A

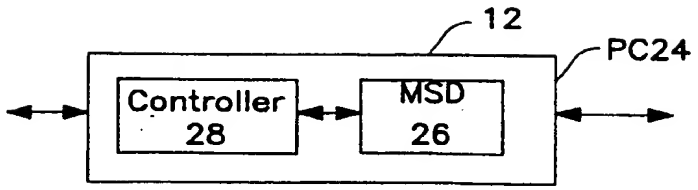

12
PC24
Controller
28
MSD
26

FIG. 2B

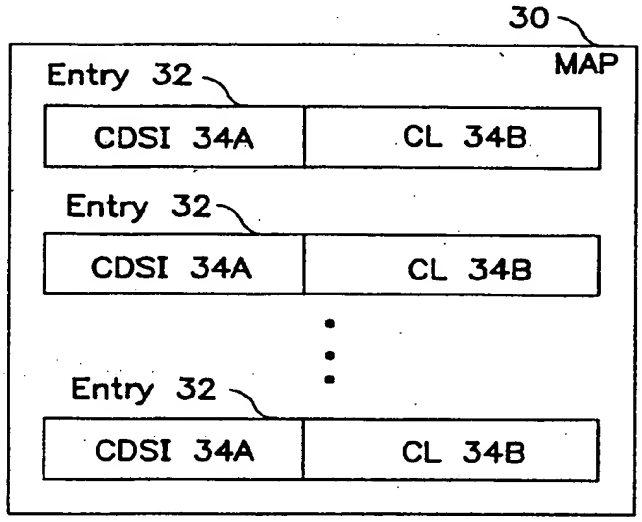

30
MAP
Entry 32
CDSI 34A
CL 34B
Entry 32
CDSI 34A
CL 34B
Entry 32
CDSI 34A
CL 34B

FIG. 2C

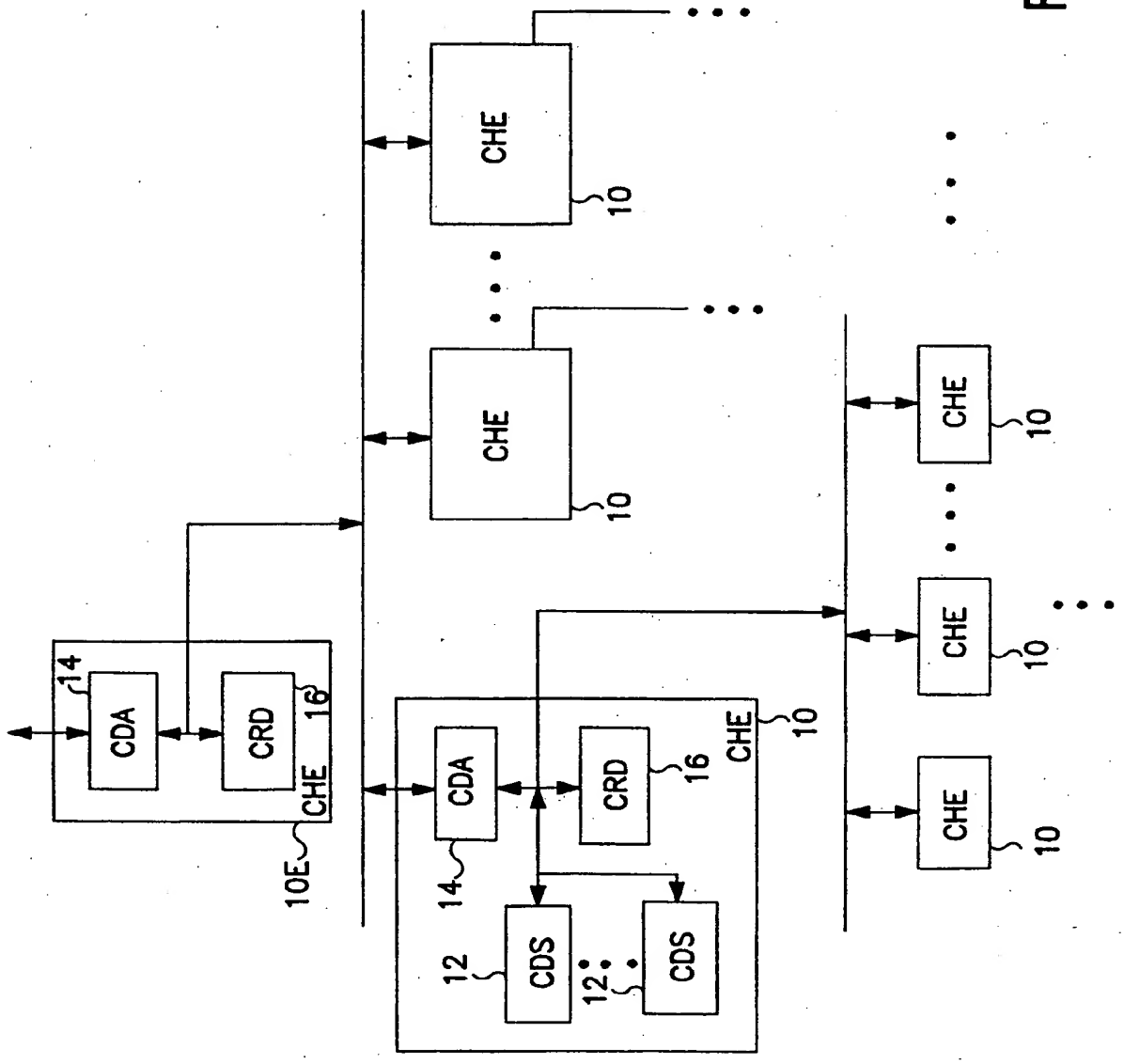


FIG. 3

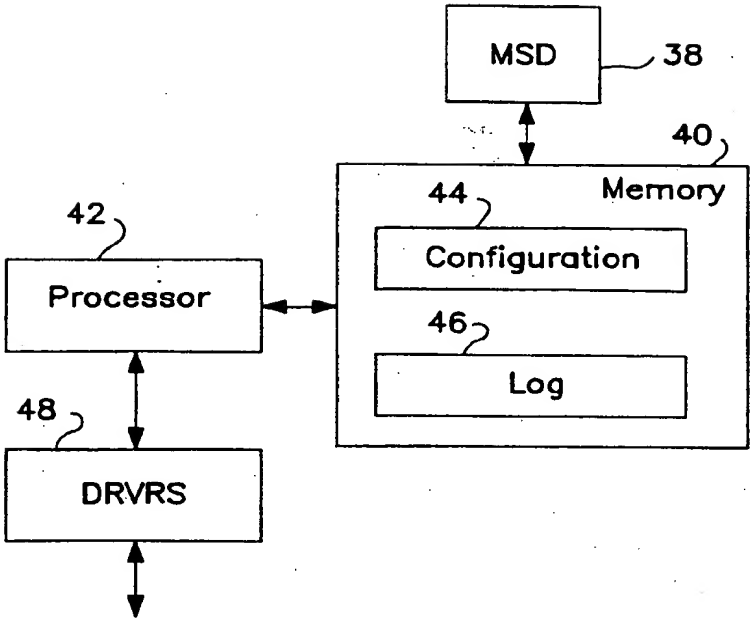

MSD
38
40
Memory
44
Configuration
46
Log
42
Processor
48
DRVRS

FIG. 4

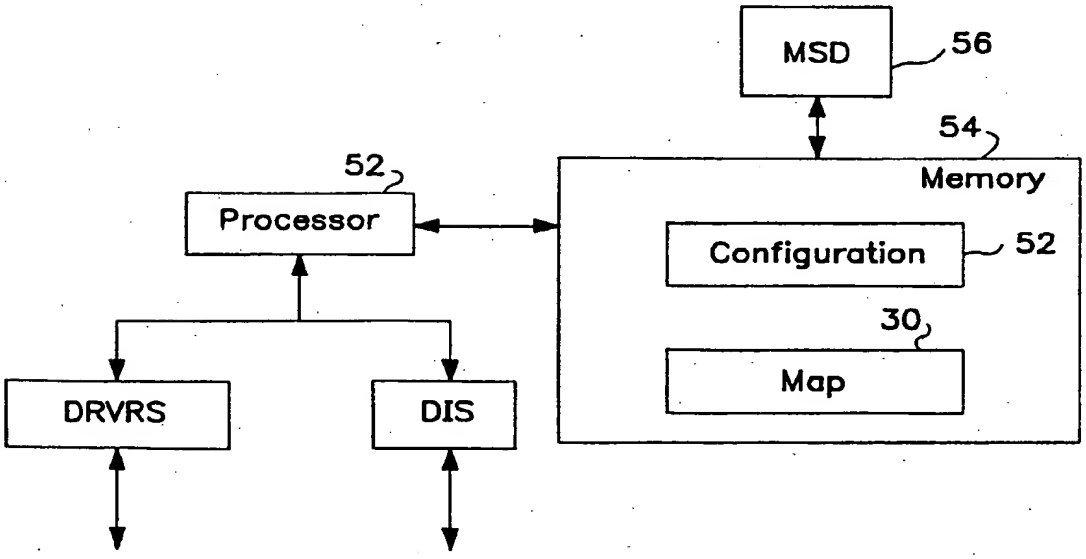

MSD
56
54
Memory
52
Processor
Configuration
52
30
Map
DRVRS
DIS

FIG. 5

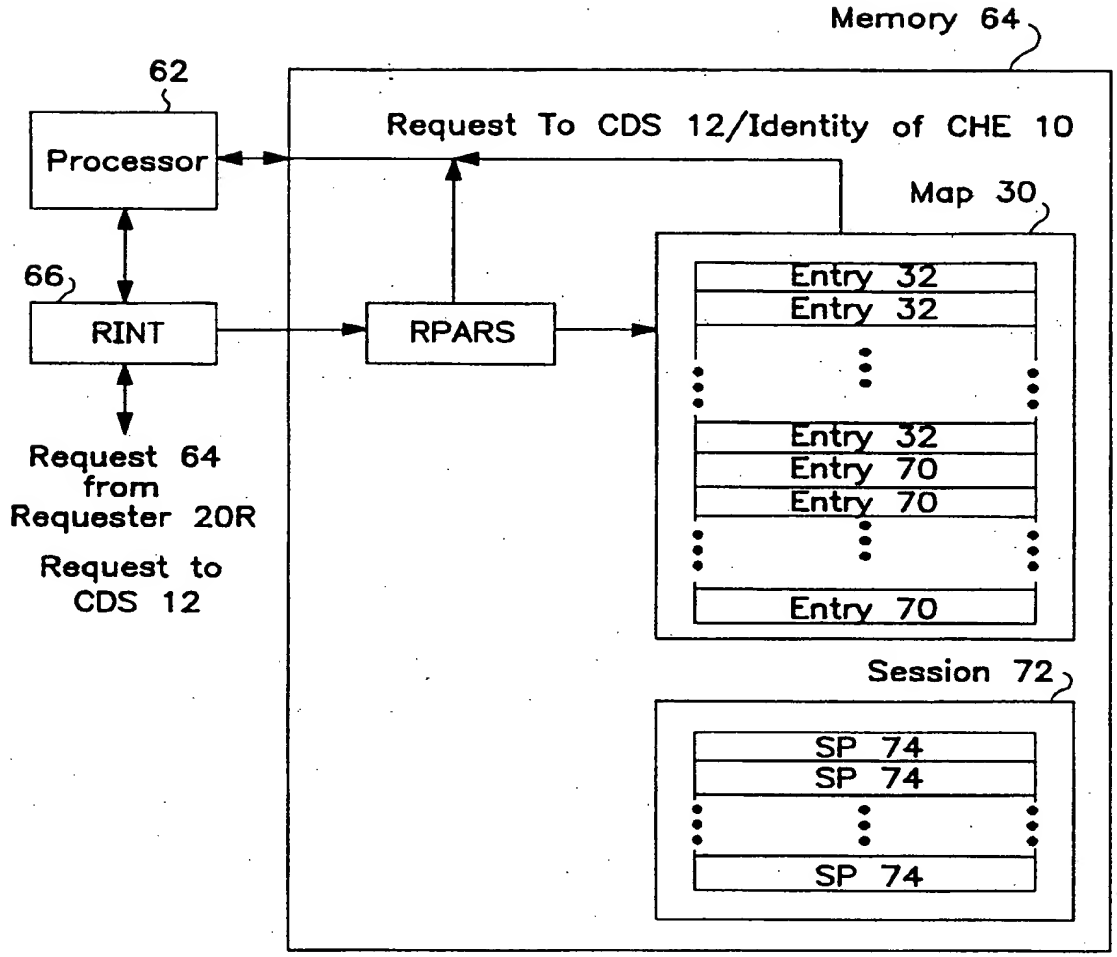

Memory 64
62
Processor
Request To CDS 12/Identity of CHE 10
Map 30
66
RINT
RPARS
Entry 32
Entry 32
Entry 32
Entry 70
Entry 70
Entry 70
Request 64 from Requester 20R
Request to CDS 12
Session 72
SP 74
SP 74
SP 74

FIG. 6

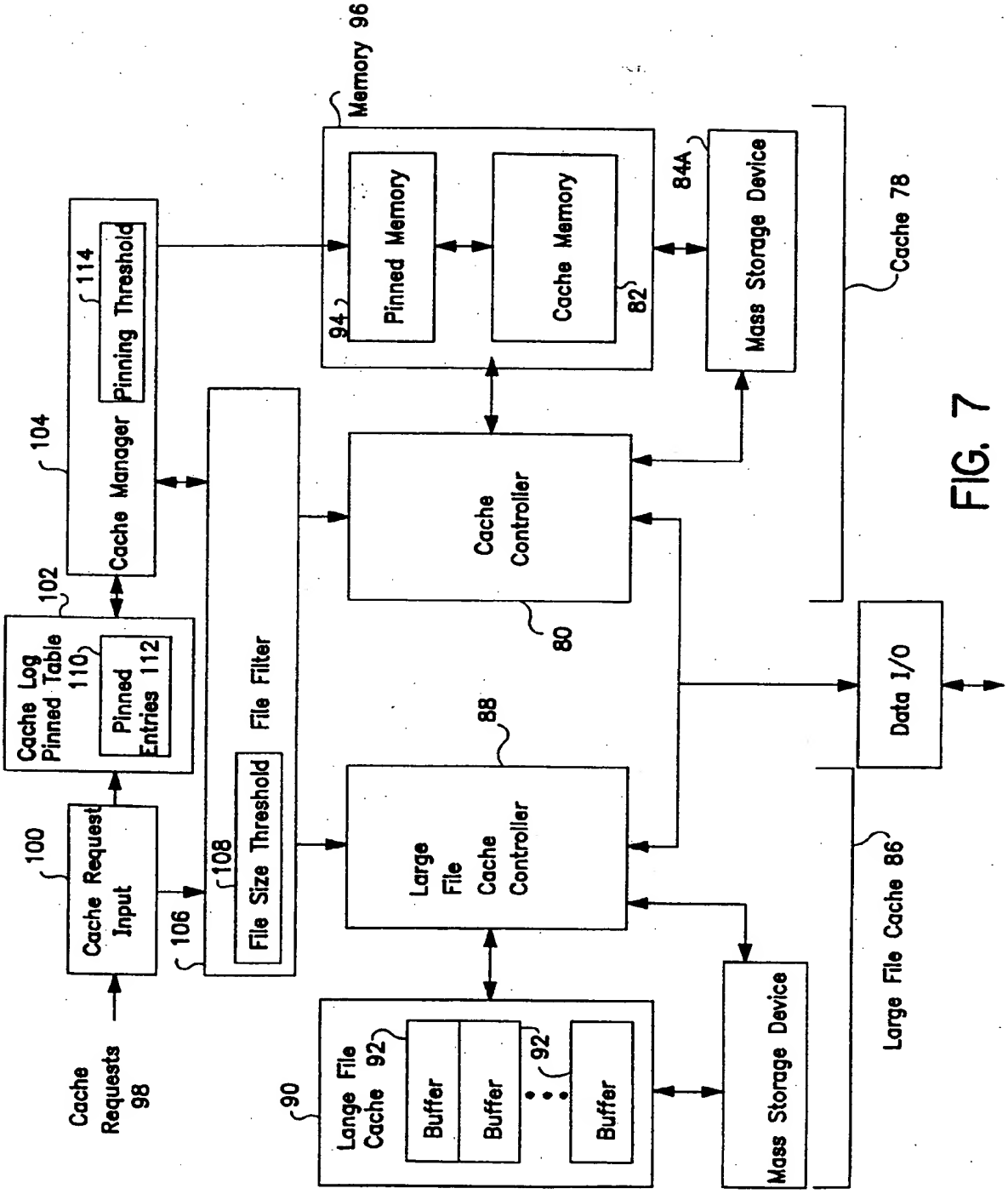

Memory 96
Pinned Memory
Cache Memory
94
82
84A
Mass Storage Device
Cache 78
Cache Manager
Pinning Threshold
114
104
Cache Log
Pinned Table
110
Pinned
Entries 112
102
Cache
Controller
80
File Filter
File Size Threshold
108
106
Cache Request
Input
100
Large
File
Cache
Controller
88
Data I/O
Cache
Requests
98
Lange File
Cache 92
90
Buffer
Buffer
92
Buffer
Mass Storage Device
Large File Cache 86

FIG. 7